

# Tree-structure Expectation Propagation for Decoding LDPC codes over Binary Erasure Channels

Pablo M. Olmos, Juan José Murillo-Fuentes  
Departamento de Teoría de la Señal y Comunicaciones.  
Universidad de Sevilla.  
email: {olmos, murillo}@us.es

Fernando Pérez-Cruz  
Departamento de Teoría de la Señal y Comunicaciones.  
Universidad Carlos III Madrid.  
email: fernando@tsc.uc3m.es

**Abstract**—Expectation Propagation is a generalization to Belief Propagation (BP) in two ways. First, it can be used with any exponential family distribution over the cliques in the graph. Second, it can impose additional constraints on the marginal distributions. We use this second property to impose pair-wise marginal distribution constraints in some check nodes of the LDPC Tanner graph. These additional constraints allow decoding the received codeword when the BP decoder gets stuck. In this paper, we first present the new decoding algorithm, whose complexity is identical to the BP decoder, and we then prove that it is able to decode codewords with a larger fraction of erasures, as the block size tends to infinity. The proposed algorithm can be also understood as a simplification of the Maxwell decoder, but without its computational complexity. We also illustrate that the new algorithm outperforms the BP decoder for finite block-size codes.

## I. INTRODUCTION

In this paper we propose a new algorithm to decode low-density parity-check (LDPC) codes over the binary erasure channel (BEC). The proposed algorithm has a better performance than the traditional Belief Propagation (BP) decoder with the same computational complexity. The analysis of the BP decoder over independent and identically distributed BEC with erasure probability  $\epsilon$  has been detailed in [1], [2], [3], in which its limiting performance and optimization are addressed. Once we represent the LDPC code by the bipartite graph induced by its parity-check matrix (Tanner Graph [4]), the BP decoding algorithm can be easily described. The degree of a check node is the number of variable nodes connected to it in the bipartite graph, and vice-versa.

We initialize the decoder by removing from the graph all the variable nodes corresponding to non-erased bits. We also remove all the connections from these variable nodes. After removing a variable node whose value was one, we change the parity of the check node(s) it was connected to. After the initialization stage, the BP algorithm proceeds by removing a check node and a variable node in each step:

- 1) It looks for any check node linked to a single variable node (a check node of degree one). The BP decoder copies the parity of this check node into the variable node and removes the check node.

- 2) Second, it removes the variable node that we have just de-erased. If the variable was a one, it changes the parity of the check node(s) it was connected to.
- 3) It repeats Steps 1 and 2 until all the variable nodes have been removed, successfully finishing the decoding of the received word, or until there are no degree-one check nodes left, yielding an unsuccessful decoding.

The asymptotic analysis of the BP performance decoding LDPC codes over the BEC is derived in [1], [5] and capacity-achieving degree distributions are presented in [1], [3]. However, for graphs with cycles, the performance of the BP decoder for any finite-length code is always inferior to the maximum a posteriori (MAP) decoder. For example, in [6] it is shown that a BP decoder for a rate-1/2 regular LDPC code with three ones per column and six ones per row can decode channels with erasure probability up to  $\epsilon = 0.4294$ , which is typically denoted as the BP capacity, as the block size tends to infinity, while the limiting MAP performance reaches 0.48815, which is denoted as the MAP capacity.

In [6], the authors proposed the Maxwell decoder to achieve the MAP solution for decoding LDPC codes over BEC. Once the BP gets stuck, because there are no additional degree-one check nodes, the Maxwell decoder assumes that the bits of a set of variable nodes are known so there is one degree-one check node that allows restarting the BP algorithm. This step is repeated as many times as needed, until all variable nodes have been accounted for. The complexity of the Maxwell decoder grows exponentially with the number of guessed variables and there is no meaningful upper bound on how many we need to assume known. Hence, it is an impractical algorithm to decode LDPC codes over BEC, although it is a useful tool to examine the code performance and derive the MAP capacity [6]. The idea of a BP decoder with variable guessing was first proposed in [7] for short-length LDPC codes, where the complexity of the algorithm is relatively small.

In this paper, we propose a new algorithm that is able to continue decoding LDPC codes for BEC after an unsuccessful BP decoding. This algorithm improves the performance of the BP decoder for finite length codes at a similar complexity. We include the analysis of the asymptotic case to show that the achieved capacity is higher than the BP one. Our decoding

algorithm borrows from the Tree-structured approximations for Expectation Propagation [8] and we refer to it as the TEP algorithm. The algorithm in [8] can be understood as a generalization of the BP decoder, in the sense that the variable nodes are linked to impose some pair-wise marginal constraints, instead of trying to compute only independent marginal distributions.

The rest of the paper is organized as follows. Section II is devoted to introducing the TEP algorithm for decoding LDPC codes. In Section III, we analyze a simplified version of the TEP decoder to prove that it achieves a higher capacity than BP. In Section IV, we compare the performance of the TEP and BP decoders with finite-length regular LDPC codes. We conclude in Section VI with some final comments.

## II. TEP ALGORITHM

The TEP decoder starts once the BP gets stuck. In a way, it works as the Maxwell decoder in which some variable nodes are assumed to be known to continue decoding. But the TEP complexity is identical to the BP, as it removes a check node and a variable node in each step and its complexity does not depend on the number of assumed variable nodes.

The Tanner graph of the LDPC code, denoted as  $\mathcal{T}$ , has  $n$  variable nodes  $V_1, \dots, V_n$  and  $n(1 - R)$  check nodes  $P_1, \dots, P_{n(1-R)}$ , where  $r$  is the rate of the code. The degree of a variable node  $V_m$  and a check node  $P_j$  is, respectively, denoted by  $d_{V_m}$  and  $d_{P_j}$ . The graph  $\mathcal{T}$  is reduced by removing the non-erased variables and performing the BP decoder until there are no degree-one check nodes left. We refer to the graph at the end of the BP as  $\mathcal{T}_{BP}$ . The TEP decoder works over  $\mathcal{T}_{BP}$  using the degree-two check nodes. A check node of degree two tells us that the variable nodes connected to it are either equal, if the check has parity zero, or different otherwise. Our algorithm chooses any check node with degree two and removes it from the graph together with one of the variable nodes connected to it and the two associated edges. Then it reconnects to the remaining variable node all the check nodes that were connected to the removed variable node. Finally, the parities of the check nodes connected to the remaining variable node have to be reversed if the removed degree-two check node had parity one. We sketch the procedure in Fig. 1, where the algorithm selects the check node  $P_1$  and removes it along with the variable  $V_2$ . We can observe in Fig. 1(b) that  $V_1$  is now connected to the check nodes  $P_2$  and  $P_3$  and its degree is now  $d_{V_1} + d_{V_2} - 2$ .

The algorithm removes a check and a variable node per iteration, as the BP does. The removal of a check node and a variable does not increase the complexity of the decoder, as it happens in the Maxwell decoder [6]. Unlike the BP, the value of the removed variable node is unknown. This variable becomes known, when the remaining variable node in the graph has been decoded. Hence, the decoder only has to store a list of the removed variables and the associated remaining variables in the graph. In the example of Fig. 1, it means that  $V_2$  would be known once  $V_1$  has been de-erased.

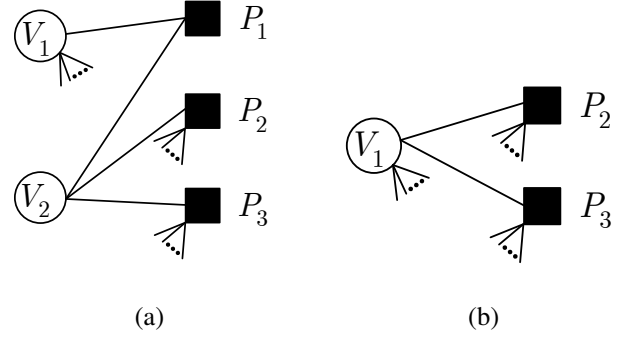


Fig. 1. In (a) we show two variable nodes,  $V_1$  and  $V_2$ , that share a check node of degree two. In (b), we show the graph once  $P_1$  and  $V_2$  have been removed. If  $P_1$  is parity one, the parities of  $P_2$  and  $P_3$  are reversed.

This procedure continues removing check and variable nodes, but it does not set any of the variables. To be able to set any variable node, we need to find a check node with degree two such that the two variable nodes connected to it also share another check node with degree three, as illustrated in Fig. 2(a). When we remove the check node  $P_3$  and the variable node  $V_2$ , the check node  $P_4$  is now degree one, as illustrated in Fig. 2(b), and we can restart the BP algorithm.

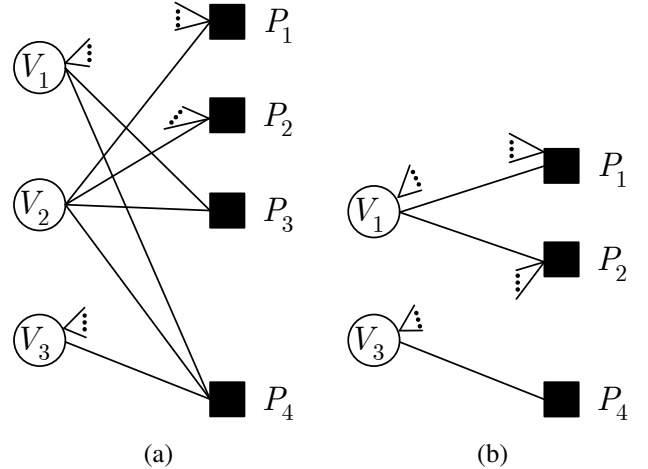


Fig. 2. In (a), the variables  $V_1$  and  $V_2$  are connected to a two-degree check node,  $P_3$ , and they also share a check node of degree three,  $P_4$ . In (b) we show the graph once the algorithm has removed  $P_3$  and  $V_2$ .

When the TEP algorithm starts running over the graph  $\mathcal{T}_{BP}$ , it is very unlikely that the two variable nodes in a check node of degree two also share a check node of degree three. However, as we remove variable and check nodes the probability of this happening grows, since we are reducing the number of nodes in the graph and increasing the degree of the remaining variable nodes. In the following section, we prove that the growth of this probability is exponential with the number of iterations of the TEP decoder. In each iteration we remove a check node of degree two and a variable node connected to it. Eventually, when a check node of degree one is created, we can restart the BP algorithm and correct error patterns undecodable by the BP decoder. The TEP decoder

halts either when the graph runs out of check nodes of degree one and two or when all the variables have been de-erased.

### III. ANALYSIS OF THE TEP DECODER

We now analyze a simplification of the above procedure and show that it achieves a higher capacity than BP. The TEP decoder first removes all the check nodes of degree two, as explained in Section II, and then it runs the BP decoder. This modification allows applying the results in [1], in which the BP capacity for BEC was computed, to analyze the TEP decoder. We consider a LDPC ensemble  $LDPC[\lambda(x), \rho(x), n]$  defined by the code length  $n$  and the edge degree distribution (DD) pair  $[\lambda(x), \rho(x)]$  [9]:

$$\lambda(x) = \sum_{i=1}^{l_{max}} \lambda_i x^{i-1}, \quad (1)$$

$$\rho(x) = \sum_{i=1}^{r_{max}} \rho_i x^{i-1}, \quad (2)$$

where  $\lambda_i$  represents the fraction of edges with left degree  $i$  in the graph and  $\rho_i$  is the fraction of edges with right degree  $i$ . The left (right) degree of an edge is the degree of the variable (check) node it is connected to. The total number of edges in the graph is denoted by  $E$  and it can be readily check that

$$E = \frac{n}{\sum_i \lambda_i / i}. \quad (3)$$

The passage of time during the decoding process is scaled so that each time unit of length  $\Delta t$  corresponds to one iteration of the decoding process as in [1]. In one iteration, the TEP removes a check node of degree two and one variable.

Let define  $l_i(t)$  and  $r_i(t)$  such that  $L_i(t) = l_i(t)E$  and  $R_i(t) = r_i(t)E$  are, respectively, the number of edges with left and right degree  $i$  in the graph at time  $t$ . We denote  $e(t)$  as the number of edges in the graph at time  $t$  divided by  $E$ . Hence,  $l_i(t)/e(t)$  for  $i = 1, \dots, l_{max}$  and  $r_i(t)/e(t)$  for  $i = 1, \dots, r_{max}$  are the coefficients of the DD pair that defines the graph at time  $t$ . In the analysis of the BP decoder in [1], the evolution of the DD pair in the limiting case, as  $n$  tends to infinity, is described by a set of differential equations. The BP decoder has analytical solution, which holds while  $r_1(t) > 0$ . In particular, the solution for  $r_1(t)$  is given by:

$$r_1(x) = \epsilon \lambda(x) [x - 1 + \rho(1 - \epsilon \lambda(x))]. \quad (4)$$

The BP capacity,  $\epsilon_{BP}$ , is the maximum probability of erasure such that  $r_1(x) > 0 \forall x \in (0, 1]$ .

#### A. TEP analysis

For a BEC with  $\epsilon > \epsilon_{BP}$ , the solution of  $r_1(x)$  in (4) touches 0 for  $x = x_{BP} > 0$  and the BP decoder gets stuck at this point. The residual graph is  $\mathcal{T}_{BP}$  and it is defined by the DD pair

$$\left[ \sum_i \frac{l_i(x_{BP})}{e(x_{BP})} x^{i-1}, \sum_i \frac{r_i(x_{BP})}{e(x_{BP})} x^{i-1}, n_{BP} \right], \quad (5)$$

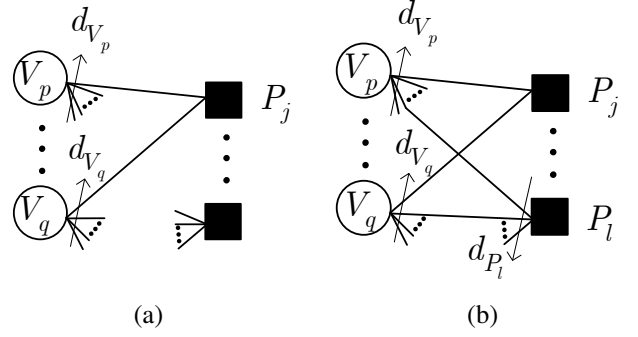


Fig. 3. Given the check node  $P_j$ , in (a) the variable nodes  $V_p$  and  $V_q$  share only the check node  $P_j$ . In (b), they share another one of degree  $m \in \{1, \dots, r_{max}\}$ .

where  $n_{BP}$  is the number of remaining variables. The TEP decoder takes the graph  $\mathcal{T}_{BP}$  as input and removes one check node of degree two and one variable per iteration. While in the BP decoder, the basic step has a unique formulation, now we have two different situations. Once a check node  $P_j$  of degree  $d_{P_j} = 2$  has been found:

- The variable nodes connected with the check,  $V_p$  and  $V_q$ , do not share another check node. See Fig. 3(a).
- $V_p$  and  $V_q$  share another check node  $P_l$  of degree  $d_{P_l} = m$ , where  $m \in \{2, \dots, r_{max}\}$ . See Fig. 3(b).

In the ensemble in (5) and provided that  $V_p$  and  $V_q$  are connected to  $P_j$  of degree two,  $p_B(x_{BP})$  represents the probability that the variable nodes also share another check node. This probability can be approximated as follows:

$$p_B(x_{BP}) \approx \sum_{m=2}^{r_{max}} \frac{(a(x_{BP}))^2 r_m(x_{BP})}{e(x_{BP})E}, \quad (6)$$

where  $a(x_{BP}) = \sum_i i l_i(x_{BP}) / e(x_{BP})$  is the average edge left degree for the ensemble in (5). This expression is proven in [10]. In the limiting case, as  $E \rightarrow \infty$ , then  $p_B(x_{BP}) \rightarrow 0$ .

We have divided the DD evolution during the decoding process in two stages, each one described by a set of differential equations:

- Stage A. The equations describe the DD evolution assuming that, in one iteration of the TEP decoder, the variable nodes connected to any check node of degree two do not share another check node.
- Stage B. The equations describe the DD evolution when, in one iteration of the TEP decoder, the variable nodes connected to a check node of degree two always share another check node.

The Stage A equations model the evolution of the DD pair at the beginning of the TEP decoding. The solution of the equations shows the probability  $p_B(t)$  grows exponentially with time. Although the Stage A equations assume  $p_B(t)$  zero or negligible, we use them to approximate the evolution of the DD pair until  $p_B(t) = 1$ . Notice that this is a conservative solution in the sense we are ignoring the check nodes of degree one that could be created before  $p_B(t)$  is one. The Stage A lasts between  $t = 0$ , when the TEP starts, and  $t = t_A$ . We

define  $t_A$  as the time in which  $p_B(t)$  is either 1 or the graph runs out of check nodes of degree two. In the first case, Stage B starts. In the second one, the TEP decoder finishes without creating check nodes of degree one and it cannot improve the BP solution.

In the Stage B,  $p_B(t) = 1$  and the variables connected to a check node of degree two also share another check node. Hence, check nodes of degree one can be created. The Stage B equations hold until the decoder removes all the check nodes of degree two. At this time, denoted as  $t_B$ , the BP decoder is run again.

### B. Stage A of decoding

Suppose that, at time  $t$ , we remove the check node  $P_j$  in Fig. 3(a), which is connected to  $V_p$  and  $V_q$ . If  $V_p$  is the remaining variable, its degree is now  $d_{V_p} + d_{V_q} - 2$ . From the edge perspective, the graph losses  $d_{V_p}$  edges with left degree  $d_{V_p}$  and  $d_{V_q}$  edges with left degree  $d_{V_q}$  and gains  $d_{V_p} + d_{V_q} - 2$  edges with left degree  $d_{V_p} + d_{V_q} - 2$ . The probability that  $d_{V_p}$  or  $d_{V_q}$  are  $i$  is  $l_i(t)/e(t)$  for  $i = 1, \dots, l_{max}(t)$ .

To derive the differential equation that describes the evolution of  $l_i(t)$  for  $i = 1, \dots, l_{max}(t)$ , we compute the expected variation of the  $L_i(t)$  distribution,  $E[L_i(t + \Delta t) - L_i(t)]$ . Hence, we are interested in cases where we have  $L_i(t + \Delta t) \neq L_i(t)$ . For  $i \geq 3$ :

- 1)  $L_i(t + \Delta t) = L_i(t) - i$  if  $(d_{V_p} = i \text{ XOR } d_{V_q} = i) \text{ AND } d_{V_p} + d_{V_q} - 2 \neq i$ .
- 2)  $L_i(t + \Delta t) = L_i(t) - 2i$  if  $d_{V_p} = d_{V_q} = i$ .
- 3)  $L_i(t + \Delta t) = L_i(t) + i$  if  $d_{V_p} \neq i, d_{V_q} \neq i \text{ AND } d_{V_p} + d_{V_q} - 2 = i$ .

The probability of each case is respectively,

$$p_{1,i} = 2 \frac{l_i(t)}{e(t)} \left( 1 - \frac{l_i(t)}{e(t)} - \frac{l_2(t)}{e(t)} \right), \quad (7)$$

$$p_{2,i} = \left( \frac{l_i(t)}{e(t)} \right)^2, \quad (8)$$

$$p_{3,i} = \sum_{q \neq i, q=1}^{i+2} \frac{l_q(t)}{e(t)} \frac{l_{i+2-q}(t)}{e(t)}. \quad (9)$$

For  $i = 2$ ,  $L_2(t + \Delta t) \neq L_2(t)$  only in Cases 1 and 3. The expected variation of the  $L_i(t + \Delta t)$  distribution with respect to  $L_i(t)$  is

$$E[L_2(t + \Delta t) - L_2(t)] = -2(p_{1,2} - p_{3,2}), \quad (10)$$

$$E[L_i(t + \Delta t) - L_i(t)] = -i(p_{1,i} + 2p_{2,i} - p_{3,i}). \quad (11)$$

If we set  $\Delta t = E^{-1}$  and  $n \rightarrow \infty$ , then (10) and (11) can be rewritten as [1]:

$$\frac{\partial l_2(t)}{\partial t} = -2(p_{1,2} - p_{3,2}), \quad i = 2, \quad (12)$$

$$\frac{\partial l_i(t)}{\partial t} = -i(p_{1,i} + 2p_{2,i} - p_{3,i}), \quad i > 3. \quad (13)$$

The initial conditions for  $t = 0$  are given by the BP ensemble defined in (5). The maximum left degree  $l_{max}(t)$  grows according to  $l_{max}(t + \Delta t) = 2l_{max}(t) - 2$  and, hence,  $l_{max}(t)$

grows as  $\mathcal{O}(2^t)$ . In each iteration, two edges of right degree two are lost. Therefore,  $E[R_2(t + \Delta t) - R_2(t)] = -2$  and the differential equation for  $r_2(t)$  is

$$\frac{\partial r_2(t)}{\partial t} = -2 \Rightarrow r_2(t) = r_2(0) - 2t = r_2(x_{BP}) - 2t. \quad (14)$$

Equations 12 and 13 represent a system of coupled nonlinear differential equations whose solution has to be computed by numerical methods. The solution shows that the average left degree  $a(t) = \sum_i i l_i(t)/e(t)$  grows exponentially with time and so does in (6).

Stage A can either finish, because it runs out of degree two check nodes or because all the variables nodes that share a check node of degree two also share another check node, which means that  $p_B(t) = 1$ . If the end time of Stage A, denoted as  $t_A$ , is  $t_{max,A} \doteq r_2(x_{BP})/2$ , then  $r_2(t)$  in (14) is zero and we cannot create new degree one-check nodes. Otherwise, if  $t_A < t_{max,A}$ , we enter Stage B, in which the decoder creates new degree-one check nodes and we can restart the BP decoder.

### C. Stage B

The graph at the beginning of this stage is defined by the solution of (12), (13) and (14) at  $t = t_A$ . We are in a situation similar to Fig. 3(b) and we select a check node as  $P_j$ . The variables  $V_p$  and  $V_q$  are also linked to  $P_l$  with degree  $d_{P_l} = m$ . If  $d_{V_p}$  and  $d_{V_q}$  are the degree of the variables, the remaining variable node has degree  $d_{V_p} + d_{V_q} - 4$ <sup>1</sup>. The check node  $P_l$  losses two edges and its degree reduces to  $d_{P_l} = m - 2$ . The equations describing the evolution of the  $l_i(t)$  distribution for  $i = 2, \dots, l_{max}(t)$  can be derived as in Section III-B. Regarding to the  $r_m(t)$  distribution, the graph loses  $m$  edges of right degree  $m$  and gains  $m - 2$  edges of right degree  $m - 2$ . As the check node  $P_l$  has degree  $d_{P_l} = m$  with probability  $r_m(t)/e(t)$ , the  $r_m(t)$  evolution can be described as follows [10]:

$$\frac{\partial r_1(t)}{\partial t} = \frac{r_3(t)}{e(t)}, \quad (15)$$

$$\frac{\partial r_2(t)}{\partial t} = 2 \left( \frac{r_4(t)}{e(t)} - \frac{r_2(t)}{e(t)} \right) - 2, \quad (16)$$

$$\frac{\partial r_m(t)}{\partial t} = m \left( \frac{r_{m+2}(t)}{e(t)} - \frac{r_m(t)}{e(t)} \right) \quad m > 2, \quad (17)$$

where the term  $-2$  in (16) represents the lost edges of the check node  $P_j$ . Let  $t_B$  represents the time when  $r_2(t) = 0$ . At  $t = t_B$ , all the check nodes of degree two have been removed. Stage B ends and the BP decoder is begins. The BP input is the DD solution of Stage B equations at  $t_B$ .

### D. TEP decoder solution for a regular LDPC ensemble

In this section, we estimate the TEP decoder capacity, denoted as  $\epsilon_{TEP}$ , for a regular LDPC ensemble with  $\lambda(x) = x^2$  and  $\rho(x) = x^5$ . The BP capacity is  $\epsilon_{BP} = 0.4294$  [1], [6]. We solve (12), (13) and (14) using Euler's method and compute  $\epsilon_{max,A} = 0.4315$ , which is the maximum channel probability

<sup>1</sup>The probability that  $V_p$  and  $V_q$  share a third check node  $P_u$  in negligible.

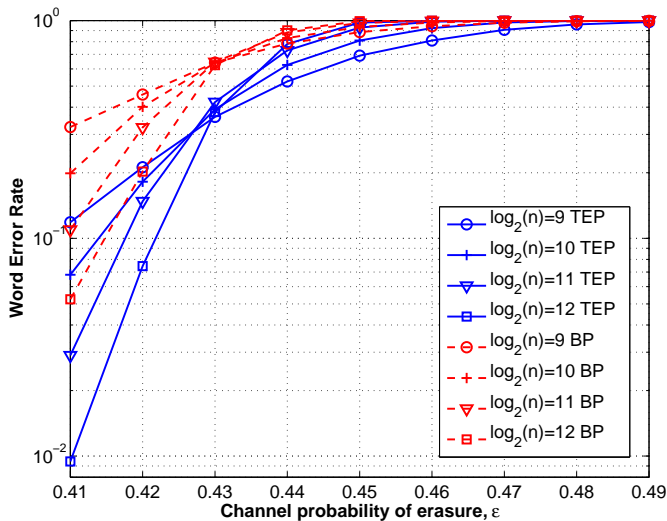


Fig. 4. WER performance of the TEP decoder (solid line) and the BP decoder (dashed line) for a regular (3,6) LDPC ensemble and code lengths  $n = 2^i$  with  $i = 9(○), 10(+), 11(▽), 12(□)$ . Each curve has been averaged for 100 different samples of the ensemble.

of erasure such that the distribution has  $p_B(t) = 1$  before the graph runs out of check nodes of degree two.

For  $\epsilon_{max,A}$  and the solution of Stage A equations at  $t_A$ , we solve the Stage B equations. The Stage B equations hold until the graph has no check nodes of degree two, which happens at  $t = t_B$ . The DD pair at  $t = t_B$  is specially favorable for the BP decoder: it has a certain number of check nodes of degree one that can be connected to variable nodes of arbitrary high degree. When the BP decoder removes one check node of degree one and a variable, the number of edges removed is very high and also the probability that new check nodes of degree one can be created. As expected, the BP solution, given the input ensemble defined by the DD at  $t = t_B$ , verifies the condition  $r_1(x) > 0 \forall x \in (0, 1]$ . Hence, the TEP algorithm for  $n \rightarrow \infty$  can decode at least up to  $\epsilon = 0.4315$ . This value is just a lower bound of the capacity of the TEP decoder, since the decoder actually creates more check nodes of degree one than the ones provided by Stage B solution. In any case, we have shown that  $\epsilon_{TEP} \geq 0.4315 > \epsilon_{BP}$ .

#### IV. EXPERIMENTAL RESULTS FOR FINITE-LENGTH CODES

In this section, we illustrate the performance of the TEP decoder, as proposed in Section II, to show that it significantly improves the performance of BP decoder for finite-length codes. We have used the regular LDPC ensemble analyzed in the above section. In Fig. IV we depict the worderror rate (WER) for the TEP decoder, solid lines, and the BP decoder, dashed lines, for code lengths  $n = 2^i$  with  $i = 9(○), 10(+), 11(▽), 12(□)$ . Each curve has been averaged for 100 different samples of the regular ensemble. The TEP decoder always improves the BP decoder.

#### V. CONCLUSIONS

In this paper, we have proposed a new decoding algorithm for LDPC codes over the BEC. The TEP algorithm exhibits a significant improvement compared to the BP decoder with identical computational complexity. The TEP algorithm borrows from the Tree-structured approximations for Expectation Propagation and it is able to continue decoding when BP halts, because there are no degree-one check nodes left. We have proved that the TEP decoder achieves a higher capacity than the BP by deriving the equations for the DD evolution along the decoding process. We have illustrated the results for regular LDPC codes. The extension of the analysis to irregular LDPC ensembles is immediate and it is carried out in [10].

#### ACKNOWLEDGEMENT

This work was partially funded by Spanish government (Ministerio de Educación y Ciencia TEC2009-14504-C02-01,02}, Consolider-Ingenio 2010 CSD2008-00010), and the European Union (FEDER).

#### REFERENCES

- [1] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Efficient erasure correcting codes," *IEEE Trans. on Information Theory*, vol. 47, pp. 569–584, 2001.
- [2] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke, "Finite length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. on Information Theory*, vol. 48, pp. 1570–1579, 2002.
- [3] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. on Information Theory*, vol. 48, no. 12, pp. 3017 – 3028, 2002.
- [4] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. on Inform Theory*, vol. 27, no. 5, pp. 533–547, 1981.
- [5] T. Richardson and R. Urbanke, "The Capacity of Low-Density Parity Check Codes under Message-Passing Decoding," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 599–618, 2001.
- [6] C. Measson, A. Montanari, and R. Urbanke, "Maxwell Construction: The Hidden Bridge Between Iterative and Maximum a Posteriori Decoding," *IEEE Trans. on Information Theory*, vol. 54, no. 12, pp. 5277–5307, 2008.
- [7] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. on Information Theory*, vol. 50, no. 3, pp. 439–454, 2004.
- [8] T. Minka and Y. Qi, "Tree-structured approximations by expectation propagation," in *Proceedings of the Neural Information Processing Systems Conference, (NIPS)*, 2003.
- [9] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Improved low-density parity-check codes using irregular graphs and belief propagation," *IEEE Trans. on Information Theory*, vol. 47, pp. 585–598, 2001.
- [10] P. M. Olmos, J. J. Murillo-Fuentes, and F. Pérez-Cruz, "Markov chain expectation propagation for decoding in erasure channels," *IEEE Trans. on Information Theory*, In preparation.